In [ ]:
```
###Lab15: 23Apr2025
#B2
```

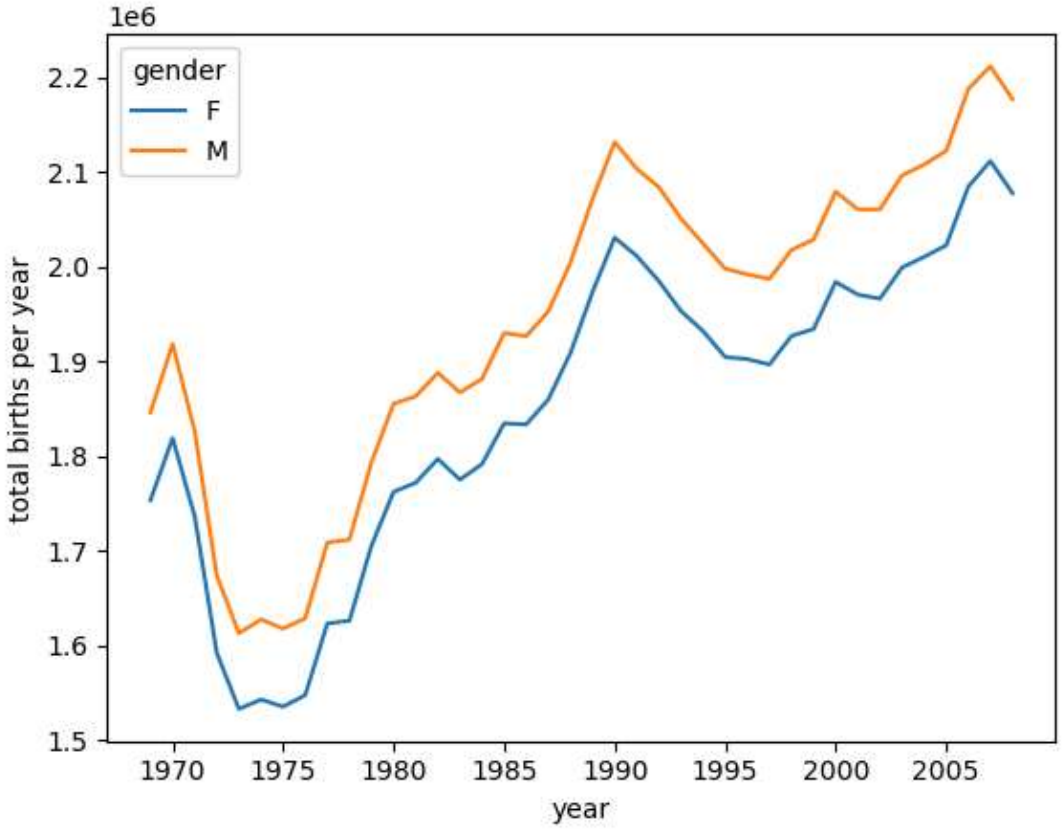
In [5]:
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np

births=pd.read_csv('births.csv')
print(births.head())
print(births.dtypes)
print(births.describe())
```

```
   year  month  day gender  births
0  1969      1  1.0      F    4046
1  1969      1  1.0      M    4440
2  1969      1  2.0      F    4454
3  1969      1  2.0      M    4548
4  1969      1  3.0      F    4548
year        int64
month       int64
day       float64
gender     object
births      int64
dtype: object
               year         month           day         births
count  15547.000000  15547.000000  15067.000000   15547.000000
mean    1979.037435      6.515919     17.769894    9762.293561
std        6.728340      3.449632     15.284034   28552.465810
min     1969.000000      1.000000      1.000000       1.000000
25%     1974.000000      4.000000      8.000000    4358.000000
50%     1979.000000      7.000000     16.000000    4814.000000
75%     1984.000000     10.000000     24.000000    5289.500000
max     2008.000000     12.000000     99.000000  199622.000000
```

In [6]:
```
temp=births.pivot_table('births',index='year',columns='gender',aggfunc='sum')
pd.pivot_table(births,index='year',columns='gender',aggfunc='sum',values='births').pl
print(temp)
plt.ylabel('total births per year')
```

```
gender        F        M
year
1969    1753634  1846572
1970    1819164  1918636
1971    1736774  1826774
1972    1592347  1673888
1973    1533102  1613023
1974    1543005  1627626
1975    1535546  1618010
1976    1547613  1628863
1977    1623363  1708796
1978    1626324  1711976
1979    1705837  1793958
1980    1762459  1855522
1981    1772037  1863478
1982    1797239  1888218
1983    1775299  1867522
1984    1791802  1881766
1985    1834774  1930290
1986    1833708  1926987
1987    1860111  1953105
1988    1909210  2004583
1989    1973712  2071981
1990    2030966  2131951
1991    2011601  2103741
1992    1985118  2084310
1993    1953456  2051067
1994    1932234  2024691
1995    1904871  1998141
1996    1902664  1992210
1997    1896928  1987401
1998    1927106  2018086
1999    1934510  2028955
2000    1984255  2079568
2001    1970770  2060761
2002    1966519  2060857
2003    1999387  2096705
2004    2010710  2108197
2005    2022892  2122727
2006    2084957  2188268
2007    2111890  2212118
2008    2077929  2177227
```

Out[6]: Text(0, 0.5, 'total births per year')

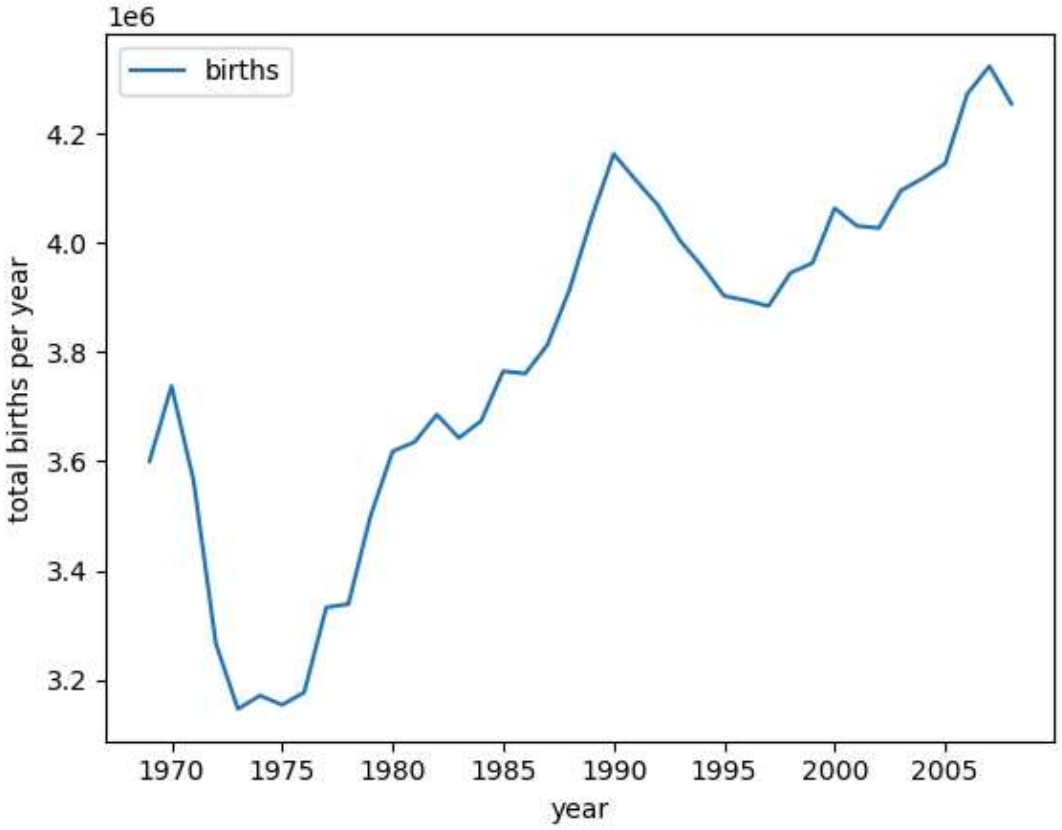

1e6
gender
F
M
2.2
2.1
2.0
1.9
1.8
1.7
1.6
1.5
total births per year
1970
1975
1980
1985
1990
1995
2000
2005
year

In [ ]:

In [13]:
```
temp = births.pivot_table(values='births', index='year', aggfunc='sum')
pd.pivot_table(births, index='year', aggfunc='sum', values='births').plot()
print(temp)
plt.ylabel('total births per year')
```

```
       births
year
1969  3600206
1970  3737800
1971  3563548
1972  3266235
1973  3146125
1974  3170631
1975  3153556
1976  3176476
1977  3332159
1978  3338300
1979  3499795
1980  3617981
1981  3635515
1982  3685457
1983  3642821
1984  3673568
1985  3765064
1986  3760695
1987  3813216
1988  3913793
1989  4045693
1990  4162917
1991  4115342
1992  4069428
1993  4004523
1994  3956925
1995  3903012
1996  3894874
1997  3884329
1998  3945192
1999  3963465
2000  4063823
2001  4031531
2002  4027376
2003  4096092
2004  4118907
2005  4145619
2006  4273225
2007  4324008
2008  4255156
```

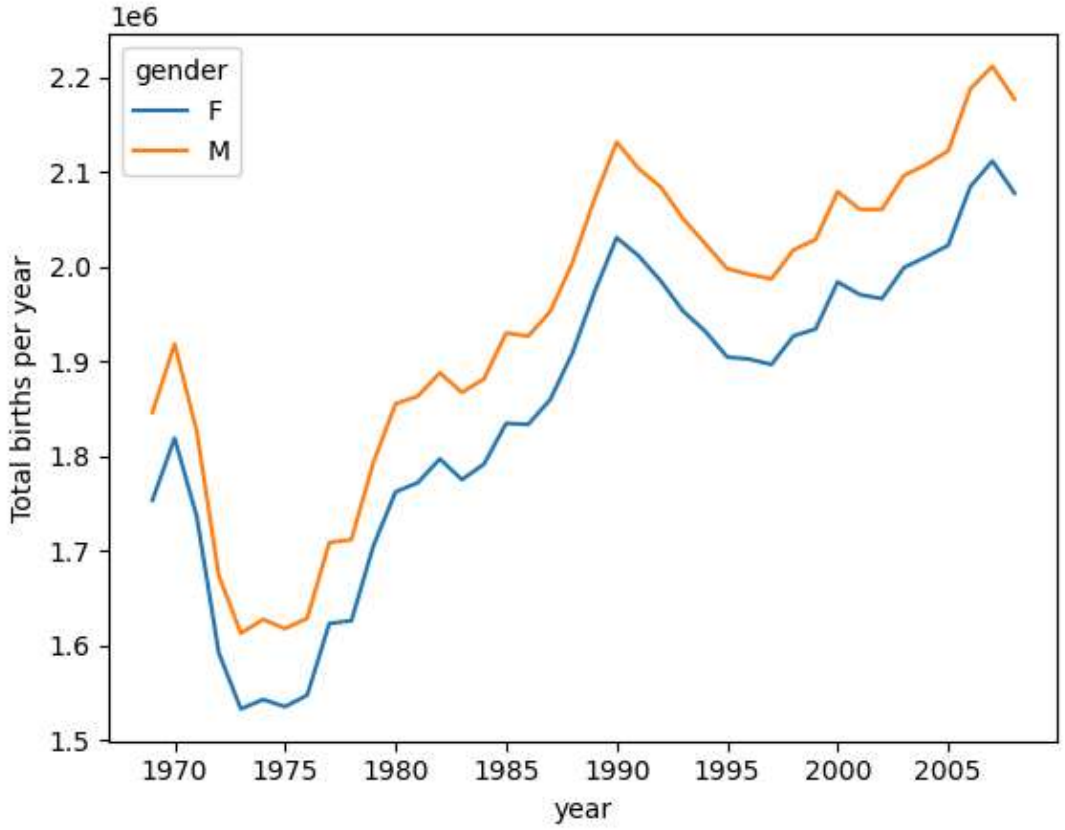
Out[13]:
```
Text(0, 0.5, 'total births per year')
```


1e6
births
4.2
4.0
3.8
3.6
3.4
3.2
total births per year
1970
1975
1980
1985
1990
1995
2000
2005
year

In [14]:
```
# Pivot table to separate male and female births per year
temp = births.pivot_table(values='births', index='year', columns='gender', aggfunc='s

# Plot the result
pd.pivot_table(births, index='year', columns='gender', values='births', aggfunc='sum'
print(temp)
plt.ylabel('Total births per year')
```

```
gender        F        M
year
1969    1753634  1846572
1970    1819164  1918636
1971    1736774  1826774
1972    1592347  1673888
1973    1533102  1613023
1974    1543005  1627626
1975    1535546  1618010
1976    1547613  1628863
1977    1623363  1708796
1978    1626324  1711976
1979    1705837  1793958
1980    1762459  1855522
1981    1772037  1863478
1982    1797239  1888218
1983    1775299  1867522
1984    1791802  1881766
1985    1834774  1930290
1986    1833708  1926987
1987    1860111  1953105
1988    1909210  2004583
1989    1973712  2071981
1990    2030966  2131951
1991    2011601  2103741
1992    1985118  2084310
1993    1953456  2051067
1994    1932234  2024691
1995    1904871  1998141
1996    1902664  1992210
1997    1896928  1987401
1998    1927106  2018086
1999    1934510  2028955
2000    1984255  2079568
2001    1970770  2060761
2002    1966519  2060857
2003    1999387  2096705
2004    2010710  2108197
2005    2022892  2122727
2006    2084957  2188268
2007    2111890  2212118
2008    2077929  2177227
```

Out[14]: Text(0, 0.5, 'Total births per year')



In [15]:
```
#count the total number of male and female births
temp = births.groupby('gender')['births'].sum()
print(temp)
```

```
gender
F    74035823
M    77738555
Name: births, dtype: int64
```

In [28]:
```
births['decade']=10 * (births['year'] //10)

births.dropna(inplace=True)
print(births)
print(type(births))

births['day']=births['day'].astype(int)
births.index=pd.to_datetime(10000* births.year + 100 * births.month + births.day, for

births['dayofweek']=births.index.dayofweek

print(births.head())
births.pivot_tables('births',index='dayofweek',columns='decade',aggfunc='mean').plot(
```

```
  Cell In[28], line 8
    births.index=pd.to_datetime(10000* births.year + 100 * births.month + births.da
y, format='%Y%m%d)

^
SyntaxError: unterminated string literal (detected at line 8)
```

In [29]:
```
births['decade'] = 10 * (births['year'] // 10)

births.dropna(inplace=True)
print(births)
print(type(births))

births['day'] = births['day'].astype(int)
births.index = pd.to_datetime(10000 * births.year + 100 * births.month + births.day,

births['dayofweek'] = births.index.dayofweek

print(births.head())
births.pivot_table('births', index='dayofweek', columns='decade', aggfunc='mean').plo
```

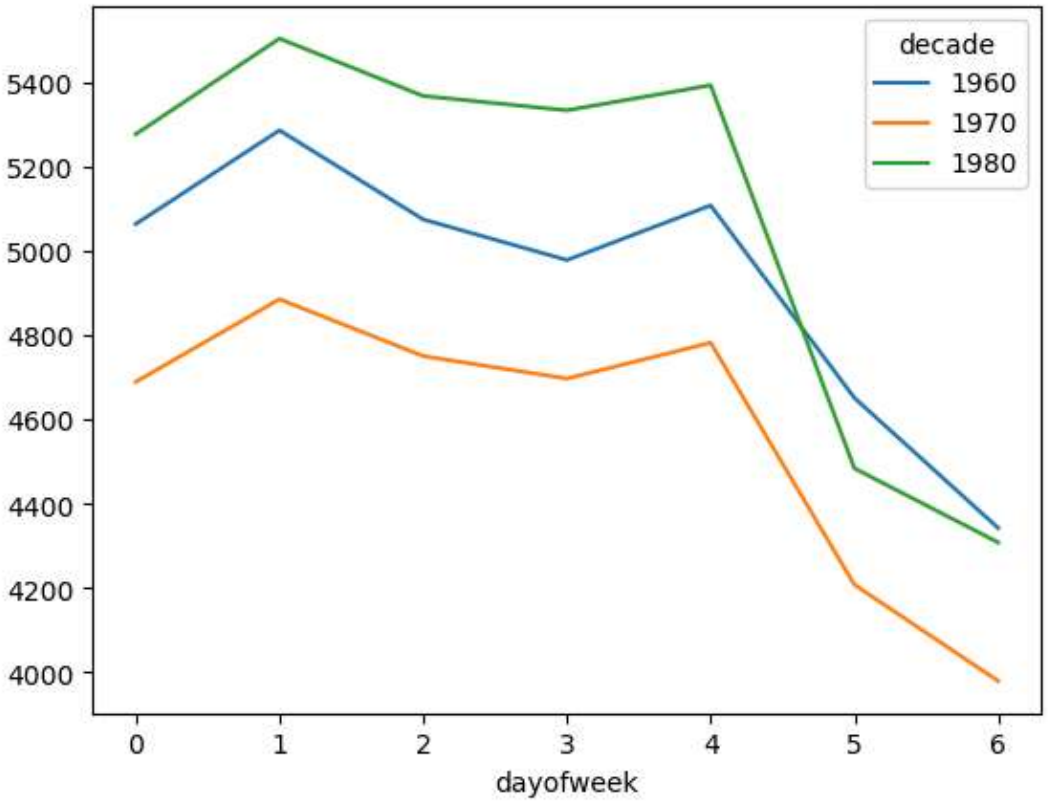
```
       year  month  day gender  births  decade
0      1969      1    1      F    4046    1960
1      1969      1    1      M    4440    1960
2      1969      1    2      F    4454    1960
3      1969      1    2      M    4548    1960
4      1969      1    3      F    4548    1960
...     ...    ...  ...    ...     ...     ...
15062  1988     12   29      M    5944    1980
15063  1988     12   30      F    5742    1980
15064  1988     12   30      M    6095    1980
15065  1988     12   31      F    4435    1980
15066  1988     12   31      M    4698    1980

[15067 rows x 6 columns]
<class 'pandas.core.frame.DataFrame'>
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[29], line 8
      5 print(type(births))
      7 births['day'] = births['day'].astype(int)
----> 8 births.index = pd.to_datetime(10000 * births.year + 100 * births.month + bir
ths.day, format='%Y%m%d')
     10 births['dayofweek'] = births.index.dayofweek
     12 print(births.head())

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1064,
in to_datetime(arg, errors, dayfirst, yearfirst, utc, format, exact, unit, infer_dat
etime_format, origin, cache)
   1062             result = arg.tz_localize(tz)
   1063 elif isinstance(arg, ABCSeries):
-> 1064     cache_array = _maybe_cache(arg, format, cache, convert_listlike)
   1065     if not cache_array.empty:
   1066         result = arg.map(cache_array)

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:229,
in _maybe_cache(arg, format, cache, convert_listlike)
    227 unique_dates = unique(arg)
    228 if len(unique_dates) < len(arg):
--> 229     cache_dates = convert_listlike(unique_dates, format)
    230     # GH#45319
    231     try:

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:430,
in _convert_listlike_datetimes(arg, format, name, tz, unit, errors, infer_datetime_f
ormat, dayfirst, yearfirst, exact)
    427         format = None
    429 if format is not None:
--> 430     res = _to_datetime_with_format(
    431         arg, orig_arg, name, tz, format, exact, errors, infer_datetime_forma
t
    432     )
    433     if res is not None:
    434         return res

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:538,
in _to_datetime_with_format(arg, orig_arg, name, tz, fmt, exact, errors, infer_datet
ime_format)
    535         return _box_as_indexlike(result, utc=utc, name=name)
    537 # fallback
--> 538 res = _array_strptime_with_fallback(
    539     arg, name, tz, fmt, exact, errors, infer_datetime_format
    540 )
    541 return res

File C:\ProgramData\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:473,
in _array_strptime_with_fallback(arg, name, tz, fmt, exact, errors, infer_datetime_f
ormat)
    470 utc = tz == "utc"
    472 try:
--> 473     result, timezones = array_strptime(arg, fmt, exact=exact, errors=errors)
    474 except OutOfBoundsDatetime:
    475     if errors == "raise":

File C:\ProgramData\anaconda3\lib\site-packages\pandas\_libs\tslibs\strptime.pyx:15
6, in pandas._libs.tslibs.strptime.array_strptime()
```

```
ValueError: unconverted data remains: 9
```

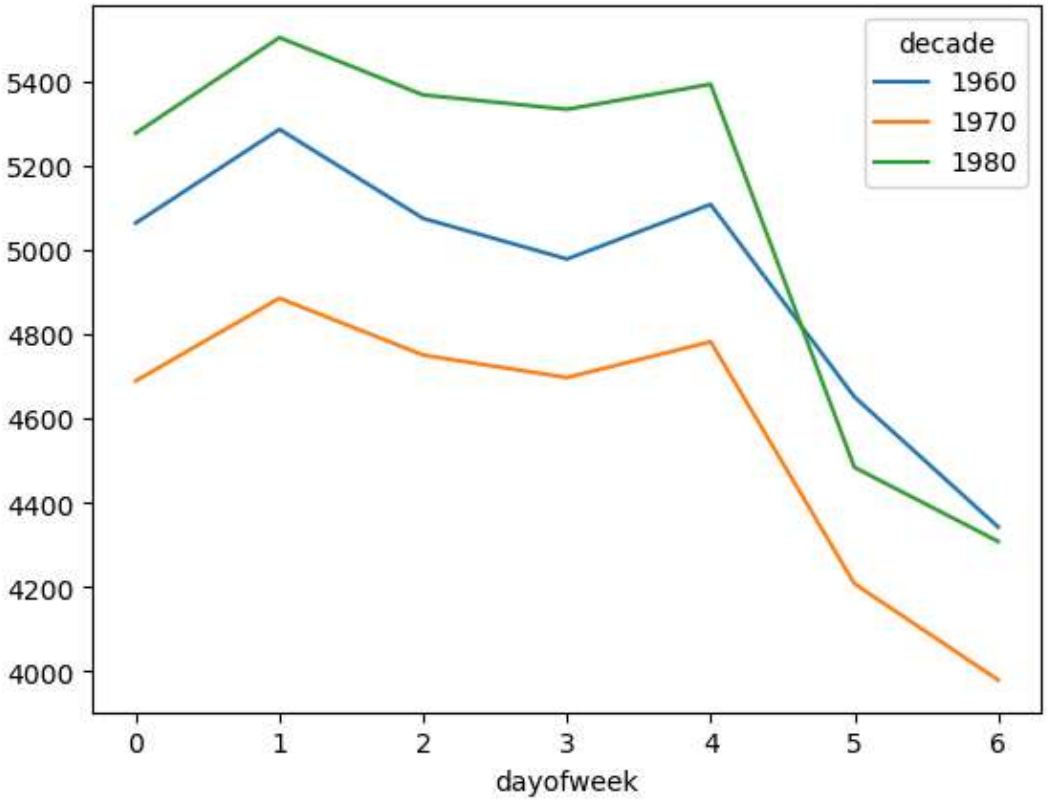
In [33]:
```
births['decade'] = 10 * (births['year'] // 10)

births.dropna(subset=['year', 'month', 'day'], inplace=True)
births['day'] = births['day'].astype(int)

births['date'] = pd.to_datetime(births[['year', 'month', 'day']], errors='coerce')
births.dropna(subset=['date'], inplace=True)

births['dayofweek'] = births['date'].dt.dayofweek

births.pivot_table('births', index='dayofweek', columns='decade', aggfunc='mean').plo
```

Out[33]: <Axes: xlabel='dayofweek'>



In [34]:
```
births['decade'] = 10 * (births['year'] // 10)

births['date'] = pd.to_datetime(births[['year', 'month', 'day']])
births['dayofweek'] = births['date'].dt.dayofweek

births.pivot_table('births', index='dayofweek', columns='decade', aggfunc='mean').plo
```

Out[34]: <Axes: xlabel='dayofweek'>



In [36]:
```
births['decade'] = 10 * (births['year'] // 10)

births['date'] = pd.to_datetime(births[['year', 'month', 'day']])
births['dayofweek'] = births['date'].dt.dayofweek


births['dayofweek'] = pd.Series(day_names)[births['dayofweek']].values

births.pivot_table('births', index='dayofweek', columns='decade', aggfunc='mean').plo
```

Out[36]: <Axes: xlabel='dayofweek'>